ARMY RESEARCH LABORATORY

Upgrades to the Closed Bomb Facility and Measurement of Propellant Burning Rate

by Tyler Wagner, John Ritter, and Barrie E. Homan

ARL-TR-5058 **January 2010**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-TR-5058 **January 2010**

Upgrades to the Closed Bomb Facility and Measurement of Propellant Burning Rate

Tyler Wagner, John Ritter, and Barrie E. Homan
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
January 2010	Final	June–July 2009

4. TITLE AND SUBTITLE
Upgrades to the Closed Bomb Facility and Measurement of Propellant Burning Rate

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Tyler Wagner, John Ritter, and Barrie E. Homan

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-WMB-D
Aberdeen Proving Ground MD 21005-5066

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-5058

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Propellants are a class of materials designed to produce low molecular weight gases that can be used in guns, missiles, and pyrotechnics. The burn rate of a propellant is used to design such munitions systems. The preferred method for obtaining the burn rate is the constant volume closed bomb. The pressure-time history obtained in closed bomb experiments, coupled with a propellant's thermochemistry and geometry, is used to calculate the linear burn rate. The hardware and software previously used to record the data needed to calculate the linear burn rate were outdated. The software that controlled the closed bomb experiments required modification in order to integrate the command set of the new Agilent oscilloscope. The new program sends user-defined commands, reads data from an external voltage source, and arranges that data in the proper format for XLCB, the burn rate data reduction code written in house using Visual BASIC in Microsoft Excel. The new code improves efficiency by creating buffers, eliminating extraneous subroutines. The new software has been tested for accuracy against the previous program using a split signal and closed bomb tests. This report presents a brief overview of closed bomb testing, a detailed account of the program and programming process, and results confirming the accuracy of the new program.

15. SUBJECT TERMS
Closed bomb, propellant characterization, burning rate

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Tyler Wagner
Unclassified	Unclassified	Unclassified	UU	38	19b. TELEPHONE NUMBER (*Include area code*) (410) 306-0709

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

Acknowledgments

We would like to acknowledge the mentorship of Dr. Kevin McNesby.

INTENTIONALLY LEFT BLANK.

1. Introduction

Propellants are energetic materials used to create thrust. They are employed in a variety of fields, from ballistics and rocketry to pyrotechnics. Propellants generate thrust by producing gas pressure, which can be harnessed to accelerate a projectile. Propellants differ from high explosives because of their burn rate. If an explosive has a burn rate that is slower than the speed of sound in the material, the energy release is described as a deflagration. On the other hand, if the speed of the reaction is supersonic, the material is a high explosive and detonates. The speed of sound through a given material is directly proportional to the density of the material; in other words, sound moves faster through more dense materials.

In order to test propellants, a constant-volume sealed chamber equipped with a pressure transducer serves as a controlled environment in which burn rates can be measured. When a propellant is ignited in the chamber, the transducer reports a voltage, which is proportional to pressure. This pressure versus time data is then a measure of the rate at which gases are produced. The rate of change of pressure, along with propellant thermochemistry and geometry, is used to calculate the burn rate. In propellant testing, the sealed chamber used to conduct this experiment is often referred to as a “closed bomb” and is shown in figure 1.

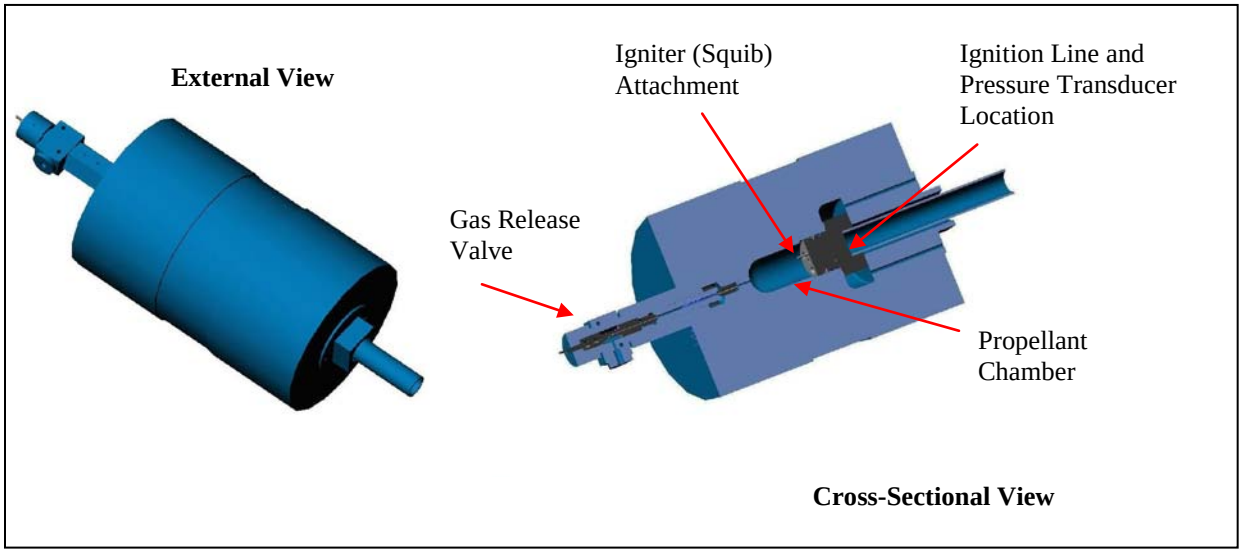

Figure 1. A 200-cc closed bomb.

2. Background

The hardware and software that is needed to read the voltage-time data produced using the pressure transducer and execute the calculations to determine the burn rate of a propellant have evolved over time. An early version of the data reduction software written to perform these tasks was BRLCB created by Oberle and Kooker (*1*) in 1993. Since then, a new program, XLCB, has been created by Homan in 2001, which updated features of BRLCB and used the capabilities of Microsoft Excel (*2*). The XLCB program works in conjunction with the Wavebook data acquisition system and XLWavebook data acquisition program (*3*) to read data from the closed bomb and calculate propellant burn rates. Unfortunately, the data acquisition hardware used to capture the pressure data was antiquated. The previous hardware (IOTech, WaveBook 512) only works with Windows 98, which is no longer widely available. An update to the hardware/software system was required to ensure continued operation of the system and decrease the processing time necessary to make the required calculations. To perform this update, an Agilent oscilloscope was purchased and a program was written (XLAgilent) to acquire data using the new oscilloscope and integrate it into the XLCB data reduction program.

3. Approach

The first step in creating a new program to control the Agilent oscilloscope was to ensure the scope was functional and able to communicate with a laptop running Windows XP. To accomplish this, a program was written in Visual BASIC (*4*) to input commands into the scope. These commands control data acquisition settings, read voltage-time data from an external source, and display the voltage-time graph onscreen. The scope was connected to the laptop via a local area network (LAN) connection. An external voltage source was connected to the oscilloscope in order to simulate the voltage response produced by the pressure transducer in a closed bomb test. With an external voltage source, in this case a simple pulse generator, the user could define parameters, such as pulse width and voltage level, to ensure the program accurately read the voltage-time curve. Initially, the Agilent control program designed to control data acquisition from the closed bomb test procedure was written in three subroutines: Initialize, Acquire, and Capture. These subroutines are described in detail in section 4.

4. Program Operation

The Agilent program began with the Initialize subroutine, which established a connection between the computer and the oscilloscope, using the Virtual Instrument Systems Architecture Communications (VISA COM) library (*5*) as the primary medium for communication. The program then wrote strings of commands to the scope declaring data collection settings, such as voltage range, time base, and trigger level. At first, these settings were inherent in the code of the program and could only be changed by rewriting the code. Eventually, all of the necessary variables were integrated into one of the XLAgilent user forms such that values could be entered or changed by the user without looking at the code. The Acquire subroutine sent additional commands to the scope determining the input channel, acquire type, and counts. The last and most intricate part of the Agilent program, the Capture subroutine, recorded and displayed the data. To verify the data were successfully collected, three measures of displaying the data were written into the program. The first used the instrument-Web interface (*6*) to graph the data, scaling the graph based on the data collection settings entered earlier. A screen capture of this interface receiving a signal from the pulse generator is shown in figure 2, with the pertinent values labeled. The next measure to ensure data were present generated a message box, displaying the voltage-time data. Finally, a new Excel spreadsheet was created into which the data were automatically inputted and graphed.

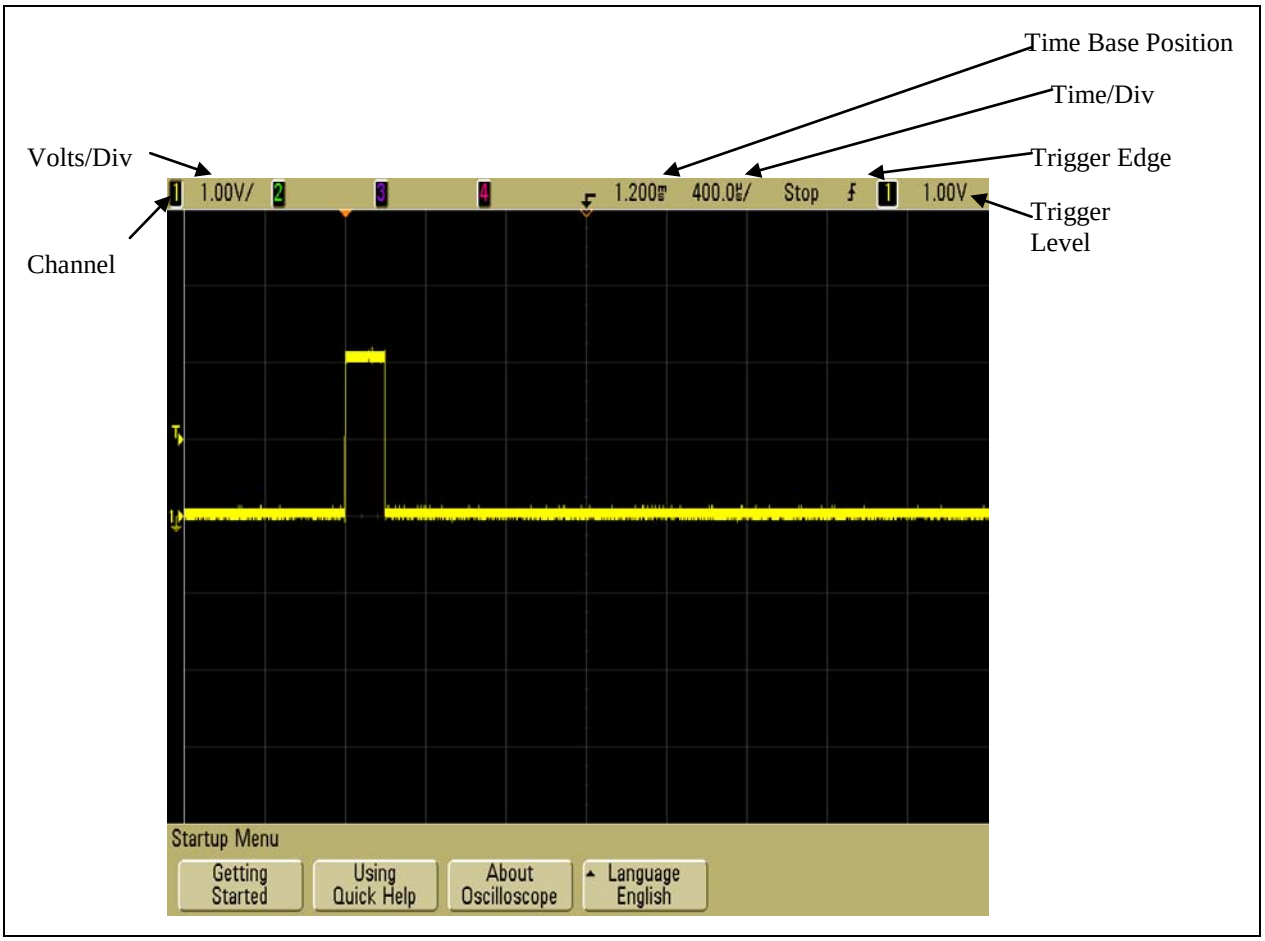

Figure 2. Instrument-web interface of a 2-V pulse.

To integrate the new XLAgilent program into XLCB, an understanding of the Wavebook command library was needed. XLWavebook used VBdaq command notation. Initially, VBdaq commands were replaced with equivalent VISA COM commands. However, the VISA COM library used by the Agilent system did not correlate to every command given by VBdaq. As a result, a bridge was constructed referring XLWavebook to the new XLAgilent program. This seemed to be an adequate solution until updates were made to the user form in which data acquisition settings were entered. Following the updates, XLAgilent would not read certain values from the user form, some values were extraneous, and some values were missing altogether. Rather than developing a mechanism to force the program to look back and forth from XLWavebook to XLAgilent, the commands from the Initiate and Acquire subroutines were combined into a new subroutine. This subroutine was placed in the Configuration user form of XLAgilent, shown in figure 3, where it could directly take the values entered into the user form, reformat them, and send them as commands to the oscilloscope. Later the Capture subroutine was integrated into the main code of XLAgilent, but changed slightly so that instead of creating a new Excel spreadsheet, it would input data values into the proper spreadsheets in XLCB for reduction. In this way, the Agilent program initially written with the three subroutines was

completely incorporated into XLAgilent without replacing any VBdaq code. The VBdaq code could now be removed as well as the three test measures to verify data were being collected, improving speed and efficiency.

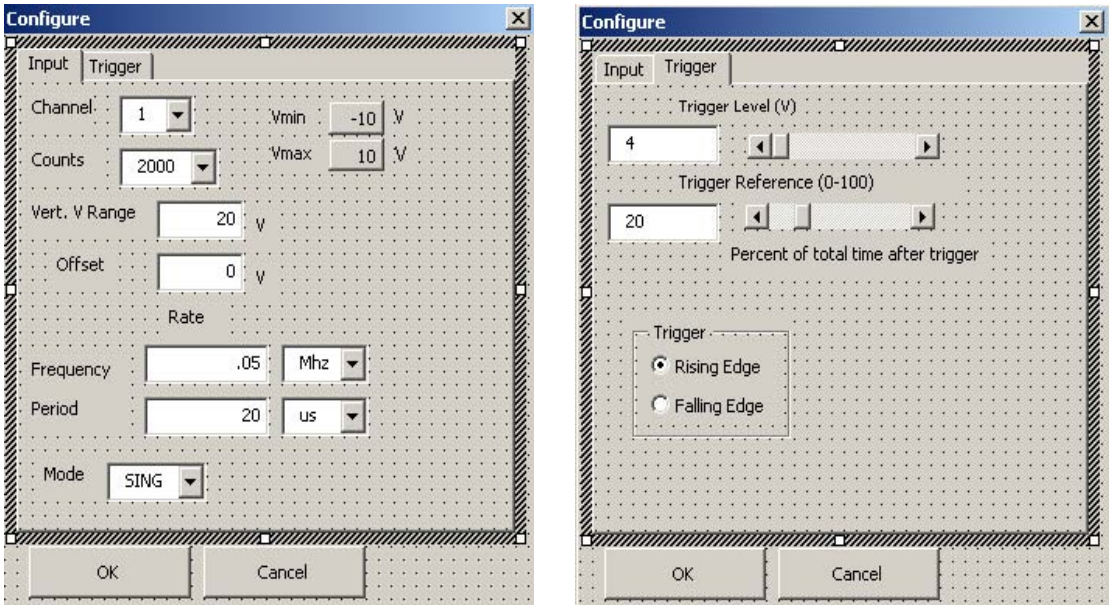

Figure 3. Configuration user form.

In order to incorporate the Initialize and Acquire subroutines, many changes had to be made to the Configuration user form and the code governing it. Even after such changes were incorporated into the user form, the XLAgilent subroutines had problems accessing the values entered in the user form. To solve this problem, a registry was created, from which the values needed to send commands to the scope could be drawn. The registry also saved previously entered values, so they would appear at the beginning of the next closed bomb test.

The performance characteristics of the Wavebook and the oscilloscope also differ, which needed to be accounted for in the software. For example, the Wavebook can only record frequencies up to 1 MHz, whereas the oscilloscope can handle frequencies up to 100 MHz. The Wavebook and oscilloscope also process different dynamic ranges, providing the oscilloscope with a longer data collection range, as shown in figure 4. One of the key changes made in the Configuration user form was the use of a gain versus voltage range. The gain used by the Wavebook correlates to volts per division in the oscilloscope, but the Agilent commands have no function to set this value. Instead, this value is calculated internally using the vertical voltage range and the offset voltage. Both values were the same, but changes needed to be made to the user form to accommodate these program differences.

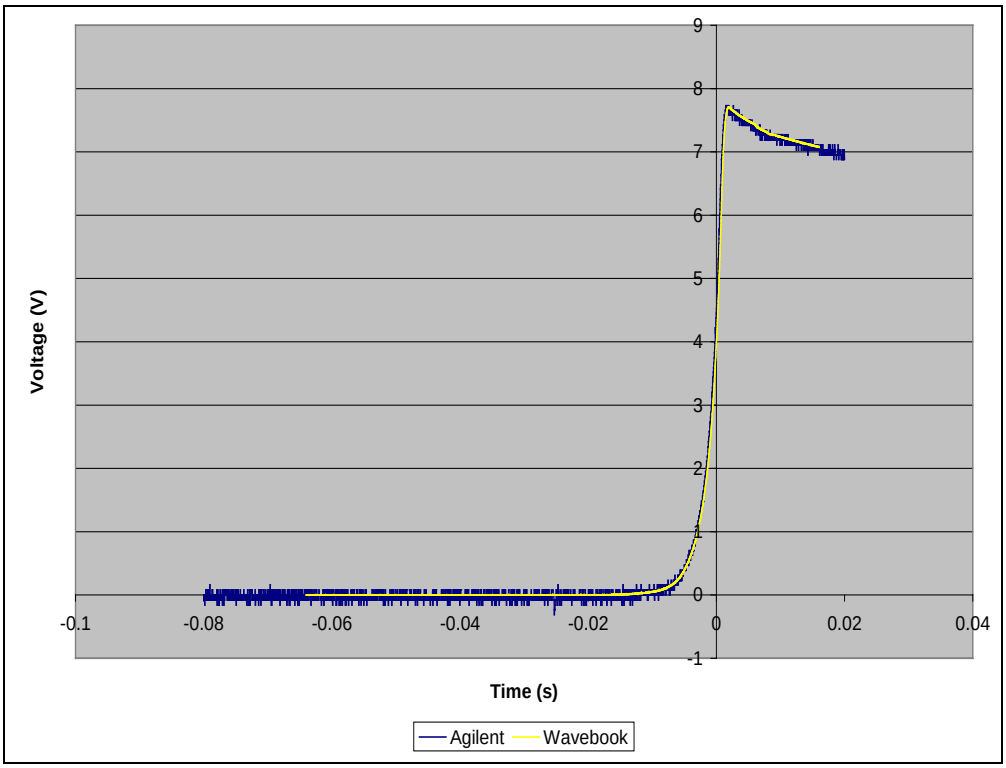

Figure 4. Voltage-time graphs (overlapped) for the JA2 test data performed on 15 July 2009.

5. Discussion

Since it is dangerous to enter the closed bomb room if a shot has not been fired, the Wavebook program implemented a wait function as a fail-safe, which would not allow the code to continue until data were recorded, thus allowing the user to know if the propellant was successfully ignited inside the bomb. The new program provided a similar fail-safe using an error message. If the user pressed "OK" on a message box querying if a shot was performed and the propellant had not ignited, then an error message popped up essentially telling the user that no data were collected. However, this was eventually deemed an inadequate fail-safe. This difference between the Wavebook and the Agilent oscilloscope was inherent in the triggering code. The Wavebook included VBdaq code that would wait for data. Once the Wavebook had the desired amount of data recorded, it would proceed with the rest of the program code. The Agilent scope had no such command. To remedy this problem XLAgilent was programmed with a status byte register (STB) query and a While loop, which continues until the trigger bit (TRG) is returned as True. TRG is bit zero with a value of one, while the rest of the bits produce even values. A simple While loop was created to query for the status byte until an odd number appeared. The program waits indefinitely for an odd number, which is only generated by a trigger, serving the same role as the VBdaq wait function.

Another improvement made to the overall program performance is how data are transferred to the Excel spreadsheet. Initially data were transferred one cell at a time. To make the program more efficient, a buffer was introduced. This buffer was already in place in the XLWavebook program, but had been commented out due to the limitations of the Wavebook. With a few modifications to the Capture subroutine, the data recorded by the oscilloscope were placed inside this buffer and could be found in one place, rather than the program going through a loop, which contained thousands of points, until all of the data were found.

6. Results

After the program was thought to be running without any problems, it was executed using a live closed bomb test with JA2, a nitrocellulose-based propellant. One problem that became apparent during calibration steps was that XLAgilent was outputting values in microseconds while XLCB was looking for data in seconds. This was remedied with a code change and testing resumed. After the first closed bomb shot, an error message appeared. The error was a formatting error, the one that appears when no data are recorded and the user tries to proceed. Per testing safety regulations, the closed bomb's valve was vented, but no gas was released. The propellant had not ignited, and the program's fail-safe had been confirmed. However, this fail-safe mechanism was ultimately replaced in favor of a mechanism that replicated the XLWavebook fail-safe. Failure to fire was attributed to an igniter issue, which was subsequently resolved. The test was carried out and the values recorded using the new program appeared to be accurate.

To confirm the accuracy of the initial test values, additional tests were performed using a split signal between the Wavebook system and the new Agilent system. The results of one of these tests are presented in figures 4–7. As shown, the data from the old Wavebook system is consistent with that of the new Agilent system, but problems have been encountered in reducing the data due to noise in the pressure-time curve. This is a hardware error rather than a software error and will be sorted out at a later date through additional testing. A grounding wire will most likely limit the noise and allow the data to be more accurately reduced. Even though noise was present in the data reduction, the heat loss observed on the Agilent system, calculated from the theoretical and observed maximum pressures, was 5.03% for the test shown in figure 4, which is acceptable. The heat loss calculated by the Wavebook was 5.83% and can be explained by the lack of noise and that the Wavebook does not accept negative values, while the Agilent system does. In either case 5–6% heat loss is reasonable for the test performed (*1*).

Notice that the voltage-time graphs align perfectly, indicating that the new program is sending the correct commands, receiving the trigger, and collecting data accurately. The remaining problem is the noise, which translates from the voltage-time graph to the pressure-time graph. The pressure derivative graph amplifies this noise, also making it apparent in the derived burn rate and vivacity graphs.

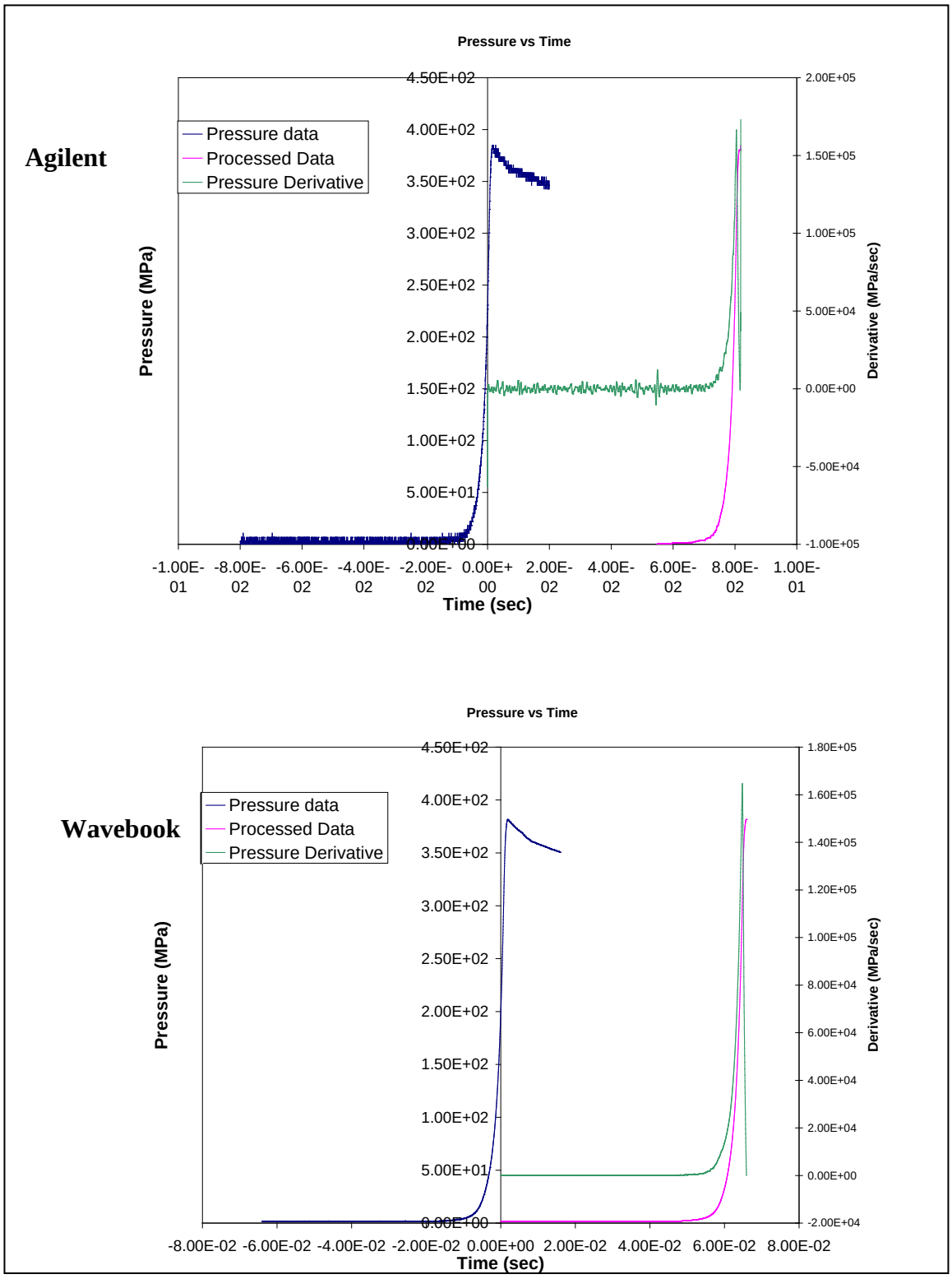

Figure 5. Pressure-time and pressure derivative graphs for the JA2 test data performed on 15 July 2009.

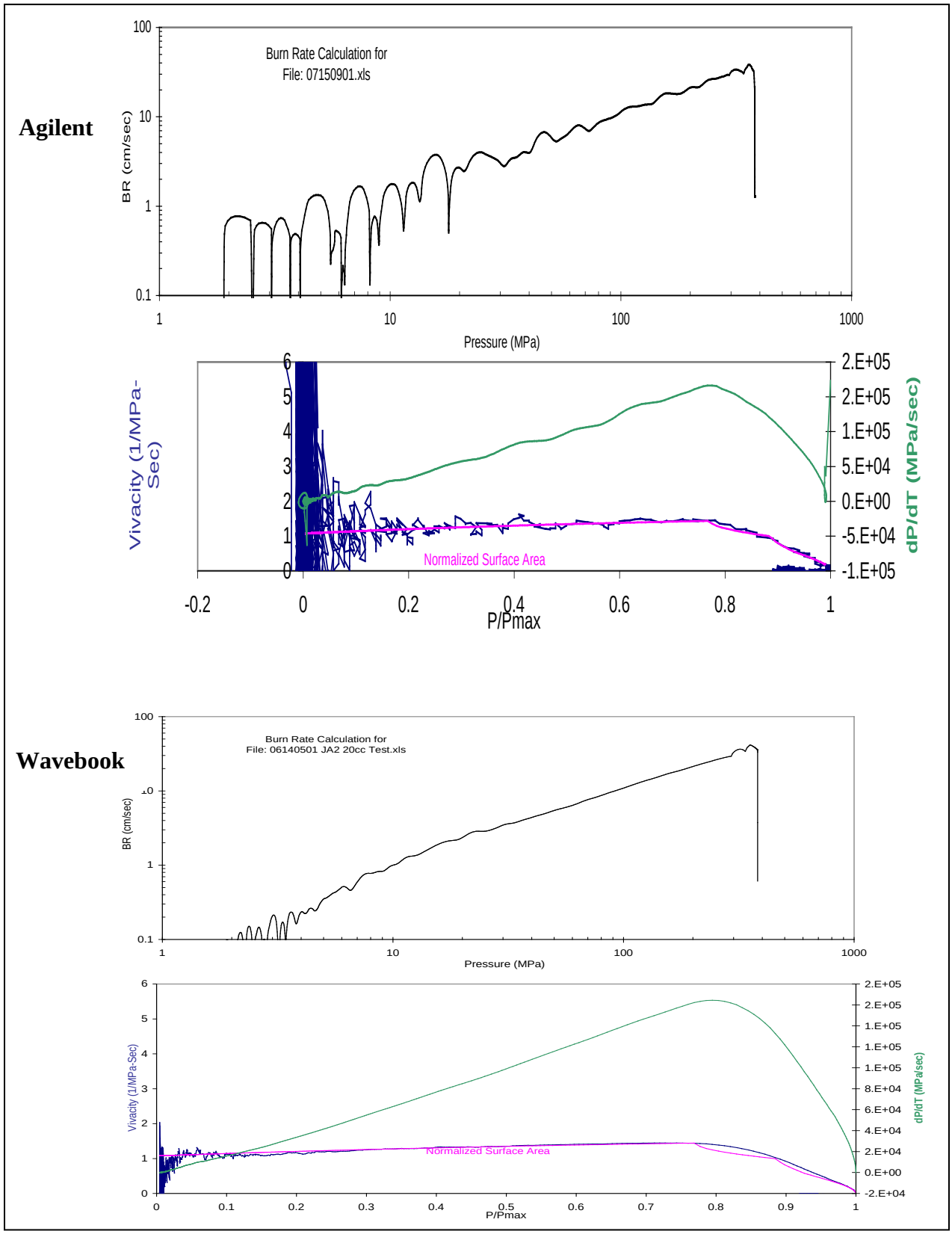

Figure 6. Burning rate, dP/dt, and vivacity graphs for the JA2 test data performed on 15 July 2009.

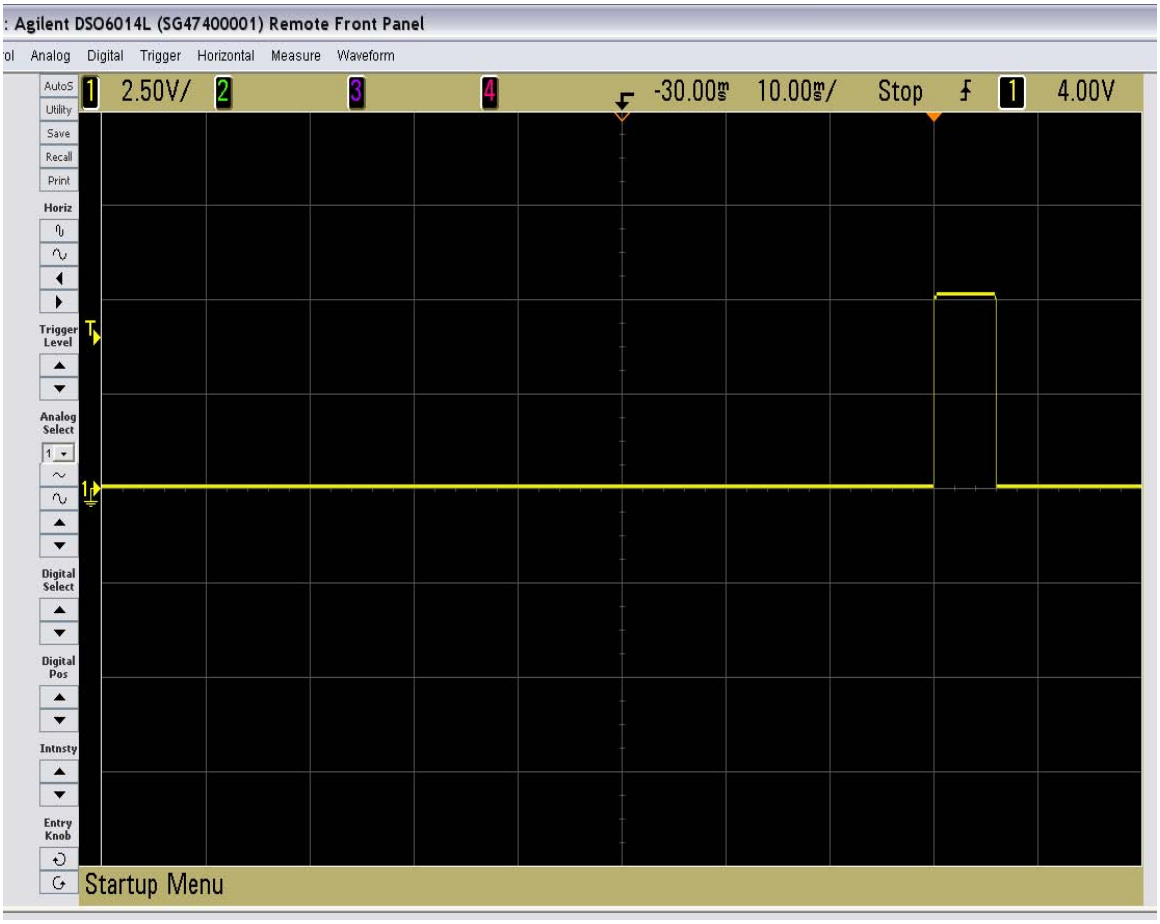

Figure 7. Instrument-Web interfaces after noise reduction for the JA2 test data performed on 15 July 2009.

7. Conclusions

The new Agilent-based data acquisition system is working and is consistent with the outdated Wavebook system. The changes incorporated in the XLAgilent program allow it to interface with the Agilent oscilloscope, as well as improve efficiency and user compatibility. With future tests and hardware improvements, noise in the data collected using the Agilent system will be reduced, and both systems (Agilent and Wavebook) should produce identical test results. After further refinement and evaluation, the Agilent oscilloscope will replace the Wavebook as the closed bomb data acquisition system.

8. Addendum

Following recent tests, the XLAgilent program has been updated. The Agilent oscilloscope has an inherent high resolution acquisition function, which smoothes incoming raw data. At the slower sweep speeds at which the program runs, this function acquires extra samples with the desired frequency and averages them in order to reduce random noise. Noise was no longer an issue after the program's acquisition mode was changed from "normal" to "high resolution." As shown in figure 7, the waveform has been smoothed compared to that shown in figure 2. Calibration tests were performed again using a split signal, and the raw data from the Agilent oscilloscope now is identical to the data from the Wavebook. It is suspected that one of the Wavebook's VBdaq functions also automatically limits noise by acting as a smoothing function. Now that matching results are being attained from the new Agilent system, the Wavebook system can be replaced by its more up-to-date counterpart.

9. References

1. Oberle, W. F.; Kooker, D. E. *BRLCB: A Closed-Chamber Data Analysis Program Part I-Theory and User's Manual*; ARL-TR-36; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, 1993.

2. Homan, B. E.; Juhasz, A. A. *XLCB: A New Closed-Bomb Data Acquisition and Reduction Program*; ARL-TR-2491; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, 2001.

3. IOtech, Inc., *Wavebook User's Manual for the Wavebook/512 System*; Cleveland, OH, 1998.

4. Boctor, D. S. *Microsoft Office Visual Basic: Step by Step*; Microsoft Press: Redmond, WA, 1997.

5. Agilent Technologies, Inc., *Agilent 6000 Series Oscilloscopes Programmer's Reference*; Colorado Springs, CO, 2007.

6. Agilent Technologies, Inc., *Agilent 6000 Series Oscilloscopes User's Guide*, (8^{th} edition); Palo Alto, CA, 2007.

Appendix A. Code for the Configuration User Form of XLAgilent

The following is the code for the Configuration user form of XLAgilent. It has been edited and commented.

```
Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Option Explicit
Dim configured As Boolean
Const MaxCounts = 65536
Dim Origin As String
Dim iAns As Integer
Dim sAns As Single
Dim UpdateFlag As Boolean
Dim CollType As String
Dim TRIGEDGESLOPE As String
Dim UNITHZ As Long
Dim VLev As Integer
Private ptyp As Integer
Private WaveBook As Integer
Const cCOUNTS = "Counts"
Const cCHANNEL = "Channel"
Const cGAIN = "Gain"
Const cPOLARITY = "Polarity"
Const cMODE = "mode"
Const cTRIGLEVEL = "TrigLevel"
Const cTRIGEDGE = "TrigEdge"
Const cSAMPLERATE = "SampleRate"
Const cTIMESCALE = "TimeScale"
Const cTRIGREF = "TrigRef"
Const cVRange = "VertVRange"
Const coffset = "offset"
Const cFreq = "Frequency"
Const cDigTrig = "Trigger Level"
Const cTrigLevelBar = "TrigLevelBar"
Dim waitvalue As Integer
---------------------------------------------------------------------------------------------------
Private Sub cbCancel_Click()
   Me.Hide
   Me.Tag = vbCancel
End Sub
---------------------------------------------------------------------------------------------------
Private Sub cbGo_Click()
   'check max count
```

```
    If Me.tbCounts > 65535 Then
        MsgBox "Maximum counts allowed is 65535", vbOKOnly, "Count Overflow"
        Me.tbCounts.SetFocus
        Exit Sub
    End If
'Changed Below from 1 MHz for Wavebook to 100 MHz for Agilent
    If Me.cbTimeScale = "MHz" And Me.tbSampleRate > 100 Then
        MsgBox "Maximum sample rate for the Agilent is 100 MHz"
        Exit Sub
    End If
    updateini
    Me.Tag = vbOK
    Me.Hide
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub cbMode_Change()
'Agilent does not accept differential mode
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub cbPeriodScale_Change()
'Choose Period Units
    If Not UpdateFlag Then
        Select Case Me.cbPeriodScale
            Case "S"
                Me.cbTimeScale = "Hz"
            Case "mS"
                Me.cbTimeScale = "kHz"
            Case Chr(181) & "S"
                Me.cbTimeScale = "MHz"
        End Select
    End If
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub cbTimeScale_Change()
'Choose Frequency Units
    If Not UpdateFlag Then
        Select Case Me.cbTimeScale
            Case "Hz"
                Me.cbPeriodScale = "S"
            Case "kHz"
                Me.cbPeriodScale = "mS"
            Case "MHz"
                Me.cbPeriodScale = Chr(181) & "S"
        End Select
    End If
End Sub
```

```
---------------------------------------------------------------------------------------------------------
Private Sub sbReference_Change()
'Choose Trigger Reference
   If Not UpdateFlag Then
      Me.tbddigitaltrigref = Me.sbReference.value
   End If
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub sbReference_Scroll()
'Scroll Bar for Trigger Reference
   Me.tbddigitaltrigref = Me.sbReference.value
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub sbTrigLevel_Change()
'Choose Trigger Level
   If Not UpdateFlag Then
      Me.tbdigtrigdisplay = Me.sbTrigLevel.value / 100
   End If
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub sbTrigLevel_Scroll()
'Scroll Bar for Trigger Level
   Me.tbdigtrigdisplay = Me.sbTrigLevel.value / 100
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub tbddigitaltrigref_Exit(ByVal Cancel As MSForms.ReturnBoolean)
'Update Trigger Reference Scroll Bar
   Me.sbReference.value = Me.tbddigitaltrigref
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub tbdigtrigdisplay_Exit(ByVal Cancel As MSForms.ReturnBoolean)
'Update Trigger Level Scroll Bar
   Me.sbTrigLevel.value = Me.tbdigtrigdisplay * 100
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub tboffset_Change()
'Updates VMax and Vmin after offset change
   tbVMax_Enter
   tbVMin_Enter
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub tbVMax_Enter()
'Calculates VMax based on vertical voltage range and offset
   If Me.tboffset.value = "" Then
      Me.tbVMax.value = CDec(Me.tbVRange.value / 2)
   Else
```

```
        Me.tbVMax.value = CDec(Me.tbVRange.value / 2) + CDec(Me.tboffset.value)
    End If
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub tbVMin_Enter()\
'Calculates VMin based on vertical voltage range and offset

     If Me.tboffset.value = "" Then
        Me.tbVMin.value = CDec(-Me.tbVRange.value / 2)
       Else
        Me.tbVMin.value = CDec(-Me.tbVRange.value / 2) + CDec(Me.tboffset.value)
    End If
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub tbVRange_Change()
'Updates VMax and Vmin after vertical voltage range change
    tbVMax_Enter
    tbVMin_Enter
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub tbSamplePeriod_Change()
'Updates Frequency after Period Change
    If val(Me.tbSamplePeriod) <> 0 Then
        Me.tbSampleRate = Format(1 / Me.tbSamplePeriod, cFLOATFORMAT)
    End If
End Sub
---------------------------------------------------------------------------------------------------------
Private Sub tbSampleRate_Change()
'Updates Period after Frequency Change
    If val(Me.tbSampleRate) <> 0 Then
        Me.tbSamplePeriod = 1 / Me.tbSampleRate
    End If
End Sub
---------------------------------------------------------------------------------------------------------
Sub update(typ As Integer)
'Where Configure code begins when directed from main XLAgilent code
    Dim i As Integer
    ptyp = typ
    If ptyp = cConfigCal Then
        ptyp = 2
        Me.Caption = "Configure Calibration Acquisition"
        CollType = cData
    Else
        ptyp = 3
        Me.Caption = "Configure Data Acquisition"
        CollType = cCalib
```

```
    End If
    UpdateFlag = True

    With tbCounts
'Establishes count value options for drop-down box in user form
        .Clear
        .AddItem 1000
        .AddItem 2000
        .AddItem 5000
        .AddItem 10000
        .value = Int(val(GetRegistry(CollType, cCOUNTS)))
'Finds last used value in registry
    End With

With cbChannel
'Establishes channel options for drop-down box in user form
        .Clear
        For i = 1 To 8
            .AddItem i
        Next i
        '.value = inisheet.Cells(3, ptyp)
        .value = Int(val(GetRegistry(CollType, cCHANNEL)))
'Finds last used value in registry
     End With

    With cbMode
'Establishes mode options for drop-down box in user form
        .Clear
'Differential command removed
'       .AddItem "DIFF"
        .AddItem "SING"
        .value = GetRegistry(CollType, cMODE)
'Finds last used value in registry
    End With

    obRising.value = GetRegistry(CollType, cTRIGEDGE)
'Finds last used value in registry
    If obRising.value = True Then
'Establishes trigger edge options
        TRIGEDGESLOPE = "POSITIVE"
    Else
        TRIGEDGESLOPE = "NEGATIVE"
    End If

    tbSampleRate.value = Int(val(GetRegistry(CollType, cSAMPLERATE)))
'Finds last used value in registry
```

```
  With cbTimeScale
'Establishes frequency units options for drop-down box in user form
      .Clear
      .AddItem "Hz"
      .AddItem "kHz"
      .AddItem "MHz"
      .value = GetRegistry(CollType, cTIMESCALE)
'Finds last used value in registry
  End With

  With cbPeriodScale
'Establishes period units options for drop-down box in user form
      .Clear
      .AddItem "S"
      .AddItem "mS"
      .AddItem Chr(181) & "S"
      Select Case Me.cbTimeScale
         Case "Hz"
            .value = "S"
         Case "kHz"
            .value = "mS"
         Case "MHz"
            .value = Chr(181) & "S"
      End Select
   End With

   If Me.cbTimeScale.value = "Hz" Then
'Converts frequency values to hertz so time values can be calculated in seconds
      UNITHZ = 1
      ElseIf Me.cbTimeScale.value = "kHz" Then
      UNITHZ = 1000
      Else
      UNITHZ = CLng(1000000)
   End If

   With tbVRange
      .value = Int(val(GetRegistry(CollType, cVRange)))
'Finds last used value in registry
   End With
   With tboffset
      .value = Int(val(GetRegistry(CollType, coffset)))
   End With
   With tbSampleRate
      .value = val(GetRegistry(CollType, cFreq))
   End With
```

```
    With tbdigtrigdisplay
       .value = Int(val(GetRegistry(CollType, cDigTrig)))
    End With
    With sbTrigLevel
       .value = Int(val(GetRegistry(CollType, cTrigLevelBar)))
    End With
    tbddigitaltrigref.value = Int(val(GetRegistry(CollType, cTRIGREF)))
    Me.sbReference.value = Int(val(tbddigitaltrigref))

    If val(Me.tbdigtrigdisplay) = 0 Then
          VLev = 0
       Else
          VLev = Int(val((Me.tbdigtrigdisplay) * 100))
    End If

       Me.tbdigtrigdisplay = VLev / 100

ufConfigure.Show
'SHOWS THE CONFIGURE USER FORM, ALLOWING THE USER TO CHANGE VALUES

With Sheets(cRAWDATA).Rows(1)
'Inputs count and offset values into designated spreadsheet cells
       .Font.Bold = True
       .Cells(, 10).value = CInt(Int(val(GetRegistry(CollType, cCOUNTS))))
       .Cells(, 7).value = Me.tboffset.value
    End With

    configured = True
    UpdateFlag = False

Initialize
'PROGRAMS THE OSCILLOSCOPE WITH VALUES DESIGNATED IN THE USER FORM

End Sub
---------------------------------------------------------------------------------------------------------
Sub UpdateLevels(BP As Boolean)
'Updates VMax and VMin levels, providing voltage range for trigger level
    Dim VLev As Integer
    If val(Me.tbdigtrigdisplay) = 0 Then
          VLev = 0
       Else
          VLev = Int(val((Me.tbdigtrigdisplay) * 100))
    End If
    Me.lbTrigLevel = "Trigger Level (" & Me.tbVMin & " to " & Me.tbVMax & ")"
End Sub
---------------------------------------------------------------------------------------------------------
```

```
Private Sub updateini()
'REGISTRY WHERE ENTERED VALUES ARE STORED FOR LATER TESTS
   SetRegistry CollType, cCOUNTS, tbCounts.value
   SetRegistry CollType, cCHANNEL, cbChannel.value
   'SetRegistry CollType, cGAIN, cbGain.value
   'SetRegistry CollType, cPOLARITY, obBipolar.value
   SetRegistry CollType, cMODE, cbMode.value
   SetRegistry CollType, cTRIGLEVEL, tbdigtrigdisplay.value
   SetRegistry CollType, cTRIGEDGE, obRising.value
   SetRegistry CollType, cTRIGLEVEL, tbdigtrigdisplay.value
   SetRegistry CollType, cSAMPLERATE, tbSampleRate.value
   SetRegistry CollType, cTIMESCALE, cbTimeScale.value
   SetRegistry CollType, cTRIGREF, tbddigitaltrigref.value
   SetRegistry CollType, cVRange, tbVRange.value
   SetRegistry CollType, coffset, tboffset.value
   SetRegistry CollType, cFreq, tbSampleRate.value
   SetRegistry CollType, cDigTrig, tbdigtrigdisplay.value
   SetRegistry CollType, cTrigLevelBar, sbTrigLevel.value
End Sub
-------------------------------------------------------------------------------------------------------
Sub Initialize()
Set myMgr = New VisaComLib.ResourceManager
   Set myScope = New VisaComLib.FormattedIO488
'Sets the VISA COM library as primary medium for communication with the oscilloscope

   Set myScope.IO = myMgr.Open("TCPIP0::169.254.254.254::inst0::INSTR")
'Connects the oscilloscope to the computer via LAN line

   myScope.IO.Clear
   myScope.WriteString "*RST"
'Clears the oscilloscope's previous settings
   myScope.WriteString ":CHAN1:PROBe 1"
'Sets the probe attenuation ratio
   myScope.WriteString (":CHAN1:RANGe " + CStr(Me.tbVRange.value))
'Sets the vertical voltage range
   myScope.WriteString (":CHAN1:OFFSet " + CStr(Me.tboffset.value))
'Sets the voltage offset
   myScope.WriteString ":CHAN1:PROB:STYP SING"
'Sets the signal type
   myScope.WriteString (":TRIG:EDGE:SOURce CHAN" + CStr(Int(val(GetRegistry(CollType,
cCHANNEL)))))
'Sets the source channel
   myScope.WriteString (":TRIG:EDGE:SLOPe " + CStr(TRIGEDGESLOPE))
'Sets the trigger slope
   myScope.WriteString (":TRIG:EDGE:LEVel " + CStr((Me.sbTrigLevel.value / 100)))
'Sets the trigger level
```

```
   myScope.WriteString ":TRIG:SWEep NORMal"
   myScope.WriteString (":TIMebase:RANGe " + Format(((Me.tbSamplePeriod / UNITHZ) *
Me.tbCounts), cFLOATFORMAT))
‘Sets the total time range
   myScope.WriteString (":TIMebase:POSition " + Format(((Me.tbddigitaltrigref - 50) / 100) *
((Me.tbSamplePeriod / UNITHZ) * Me.tbCounts), cFLOATFORMAT))
‘Sets the percentage of total time before and after the trigger occurs
   myScope.WriteString ":ACQuire:COMPlete 100"
‘Data acquisition is complete after 100 percent of data buckets are filled
   myScope.WriteString ":ACQuire:COUNt 1"
‘Changed Acquire type to high resolution to eliminate noise
 ‘   myScope.WriteString ":ACQ:TYPE NORMal"
  myScope.WriteString  ":ACQ:TYPE HRES"
  myScope.WriteString ":SINGLE"
‘Causes the instrument to stop after a single trigger of data

 ‘   MsgBox "Waiting For Trigger"
‘Makes a message box pop up. The user clicks ok after the shot is believed to have been fired. If
data has not been acquired, error 2147221439 will appear.

WAITFORTRIGGER

   CheckForInstrumentErrors
End Sub
---------------------------------------------------------------------------------------------------
Private Sub CheckForInstrumentErrors()
‘Checks for errors
On Error GoTo VisaComError
Dim strErrVal As String
Dim strOut As String

myScope.WriteString "SYSTEM:ERROR?"
strErrVal = myScope.ReadString
While val(strErrVal) <> 0
   strOut = strOut + "INST Error: " + strErrVal
   myScope.WriteString ":SYSTEM:ERROR?"
   strErrVal = myScope.ReadString
Wend

If Not strOut = "" Then
   MsgBox strOut, vbExclamation, "INST Error Message"
   myScope.FlushWrite (False)
      myScope.FlushRead
End If
Exit Sub
```

```
VisaComError:
   MsgBox "VISA COM Error:" + vbCrLf + Err.Description
‘Displays error message

End Sub
---------------------------------------------------------------------------------------------------------
Sub WAITFORTRIGGER()
‘New Subroutine; causes the program to wait for a trigger via a continuous, query and return
loop without the use of a message box. The status byte receive will only be odd if the trigger has
been received. The loop the mod function to determine if the status byte is odd. Setting the
waitvalue to 0 essentially clears the previous status byte. This subroutine acts exactly like the
VBdaq wait function of the old Wavebook.

myScope.WriteString "*CLS"
waitvalue = 0
While waitvalue Mod 2 = 0
myScope.WriteString "*STB?"
waitvalue = myScope.ReadNumber
Application.StatusBar = "Waiting For Trigger…"
Wend
End Sub
---------------------------------------------------------------------------------------------------------
```

Appendix B. Code from the Control Module of XLAgilent

The following is code from the Control module of XLAgilent. It was formerly part of the Capture subroutine of the Agilent program. It has been edited and commented.

```
Private Sub savedata(dest As String)
        Dim col As Integer
        Dim i As Integer
        Dim endrow As Integer
        Dim CurSheet As Variant

        'The following are used in the buffer
        Dim wkrange As range
        ReDim xy(1 To Sheets(cRAWDATA).Cells(1, 10).value, 1 To 2) As Double

        Set myMgr = New VisaComLib.ResourceManager
        Set myScope = New VisaComLib.FormattedIO488
        'Establishes VISA COM library

        Set myScope.IO = myMgr.Open("TCPIP0::169.254.254.254::inst0::INSTR")
        'Ensures oscilloscope is connected
        myScope.WriteString ":SYSTEM:SETUP?"
        'Queries for data acquisition settings sent to machine in Configure user form code
        varQueryResult = myScope.ReadIEEEBlock(BinaryType_UI1)
        'Reads the data acquisition settings from the oscilloscope
        Dim strPath As String
        strPath = "C:\Documents and Settings\Barrie Homan\Desktop\Closed Bomb\scope.dat"
        'Writes a data file in a designated folder that contains the settings used
        Close #1
        Open strPath For Binary Access Write Lock Write As #1
        Put #1, , varQueryResult
        Close #1

    'The following code displayed the data using the instrument-web interface, but is no
    longer needed
    'Dim byteData() As Byte
    'myScope.IO.Timeout = 15000
    'myScope.WriteString ":DISPLAY:DATA? BMP, SCREEN, COLOR"
    'byteData = myScope.ReadIEEEBlock(BinaryType_UI1)
    'strPath = "C:\Documents and Settings\Barrie Homan\Desktop\Closed
    Bomb\scope.bmp"
    'If Len(Dir(strPath)) Then
    '    Kill strPath
```

```
'End If
'Close #1
'Open strPath For Binary Access Write Lock Write As #1
'Put #1, , byteData
'Close #1
'myScope.IO.Timeout = 5000

Dim varSetupString As Variant
strPath = "C:\Documents and Settings\Barrie Homan\Desktop\Closed Bomb\scope.dat"
Open strPath For Binary Access Read As #1
Get #1, , varSetupString
Close #1
myScope.WriteIEEEBlock ":SYSTEM:SETUP ", varSetupString
myScope.WriteString ":WAVEFORM:SOURCE CHAN1"
myScope.WriteString ":WAVEFORM:POINTS:MODE RAW"
‘Determines the maximum amount of points collected depending on the setting
myScope.WriteString (":WAVEFORM:POINTS " +
CStr(Sheets(cRAWDATA).Cells(1, 10).value))
‘Determines the actual amount of points collected


Dim lngVSteps As Long
Dim intBytesPerData As Integer
myScope.WriteString ":WAVEFORM:FORMAT WORD"
lngVSteps = 65536
intBytesPerData = 2
Dim Preamble()
Dim intFormat As Integer
Dim intType As Integer
Dim lngPoints As Long
Dim lngCount As Long
Dim dblXIncrement As Double
Dim dblXOrigin As Double
Dim lngXReference As Long
Dim sngYIncrement As Single
Dim sngYOrigin As Single
Dim lngYReference As Long
Dim strOutput As String
myScope.WriteString ":WAVEFORM:PREAMBLE?"
‘Queries for the preamble, which contains horizontal and vertical scaling settings
Preamble() = myScope.ReadList
intFormat = Preamble(0)
intType = Preamble(1)
lngPoints = Preamble(2)
lngCount = Preamble(3)
dblXIncrement = Preamble(4)
```

```
dblXOrigin = Preamble(5)
lngXReference = Preamble(6)
sngYIncrement = Preamble(7)
sngYOrigin = Preamble(8)
lngYReference = Preamble(9)
‘The following was used in the message box containing the data points, displaying
volts/div, offset, sec/div, and delay
‘strOutput = ""
‘strOutput = strOutput + "Volts/Div = " + _
    ‘FormatNumber(lngVSteps * sngYIncrement / 8) + _
    ‘" V" + vbCrLf
‘strOutput = strOutput + "Offset = " + _
    ‘FormatNumber((lngVSteps / 2 - lngYReference) * _
    ‘sngYIncrement + sngYOrigin) + " V" + vbCrLf
‘strOutput = strOutput + "Sec/Div = " + _
    ‘FormatNumber(lngPoints * dblXIncrement / 10 * _
    ‘1000000) + " us" + vbCrLf
‘strOutput = strOutput + "Delay = " + _
    ‘FormatNumber(((lngPoints / 2) * _
    ‘dblXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf
myScope.WriteString ":WAV:DATA?"
‘Queries for data
'Dim i As Integer
Dim lngI As Long
Dim lngDataValue As Long
varQueryResult = myScope.ReadIEEEBlock(BinaryType_UI1)
‘Reads the data from the oscilloscope
For lngI = 0 To UBound(varQueryResult) _
    Step CInt((CInt(UBound(varQueryResult)) / CInt(Sheets(cRAWDATA).Cells(1,
    10).value)))
‘Uses the count to input the data in the spreadsheet correctly
If intBytesPerData = 2 Then
    lngDataValue = varQueryResult(lngI) * 256 + _
    varQueryResult(lngI + 1)

xy(CInt(((CInt(lngI) / CInt((UBound(varQueryResult) / Sheets(cRAWDATA).Cells(1,
    10).value))) + 1)), 1) = CDbl(((FormatNumber(((lngI / intBytesPerData - lngXReference)
    * dblXIncrement + dblXOrigin) * 1000000))) * 0.000001)
‘Inputs the time values into the buffer

xy(CInt(((CInt(lngI) / CInt((UBound(varQueryResult) /
    Sheets(cRAWDATA).Cells(1, 10).value))) + 1)), 2) =
    CDbl((FormatNumber((lngDataValue - lngYReference) * sngYIncrement + _
    sngYOrigin)))
‘Inputs the voltage values into the buffer
```

```
    'The following was used to display voltage-time data points in a message box
    Else
        lngDataValue = varQueryResult(lngI)
    End If
    strOutput = strOutput + "Data point " + _
        CStr(lngI / intBytesPerData) + ", " + _
        FormatNumber((lngDataValue - lngYReference) * sngYIncrement + _
        sngYOrigin) + " V, " + FormatNumber(((lngI / intBytesPerData - lngXReference) *
        dblXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf
    Next lngI

    Application.StatusBar = "Saving Data"
    Set CurSheet = ActiveSheet
    fasterupdate True
    With Sheets(cRAWDATA)
       .Activate
       .Cells(3, 1) = "Voltage"
       .Cells(3, 3) = "Calibration"
       .Cells(4, 1) = "Number of points"
       For i = 1 To 3 Step 2
          .Cells(5, i) = "Time (sec)"
          .Cells(5, i + 1) = "Volt"
       Next i
       If dest = cData Then
          col = 1
          .Cells(4, 2) = Sheets(cRAWDATA).Cells(1, 10).value
          .Cells(1, 1) = "Data From Wavebook on " & Date & " " & Time
       Else
          col = 3
          .Cells(4, 4) = Sheets(cRAWDATA).Cells(1, 10).value
          .Cells(2, 1) = "Calibration From WaveBook on " & Date & " " & Time
       End If
       endrow = FindLastUsedRow(cRAWDATA, 6, col)
       .range(Cells(6, col), Cells(endrow, col + 1)).Clear

    'The following was the previous way of inputting data into Excel spreadsheet before
    buffer was created, one cell at a time
       'For i = 1 To count
          'For i = 1 To Sheets(cV).Cells(1, 4).value
          'xy(i, 1) = Sheets(cV).Cells(i + 1, 2).value
          'xy(i, 2) = Sheets(cV).Cells(i + 1, 3).value
          ' 'xy(i, 1) = starttime + (i - 1) * period
          "xy(i, 2) = buffer(i) * slope + OFFSET
          'Next i
Set wkrange = range(.Cells(cStrtRow, col), .Cells(Sheets(cRAWDATA).Cells(1,
          10).value - 1 + cStrtRow, col + 1))
```

```
 wkrange = xy
 ‘Enters data values from buffer into XLCB spreadsheet
 End With
   CurSheet.Activate
   fasterupdate False
   Application.StatusBar = False
End Sub
```

INTENTIONALLY LEFT BLANK.

NO. OF COPIES	ORGANIZATION
1 ELECT	ADMNSTR DEFNS TECHL INFO CTR ATTN DTIC OCP 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
1	DARPA ATTN IXO S WELBY 3701 N FAIRFAX DR ARLINGTON VA 22203-1714
1 CD	OFC OF THE SECY OF DEFNS ATTN ODDRE (R&AT) THE PENTAGON WASHINGTON DC 20301-3080
1	ARDEC ATTN AMSTA AR WEE A K KLINGMAN BLDG B321 PICATINNY ARSENAL NJ 07806
1	US ARMY RSRCH DEV AND ENGRG CMND ARMAMENT RSRCH DEV AND ENGRG CTR ARMAMENT ENGRG AND TECHNLGY CTR ATTN AMSRD AAR AEF T J MATTS BLDG 305 ABERDEEN PROVING GROUND MD 21005-5001
1	DIR BENET WEAPONS LAB ATTN TECHL LIB WATERVLIET NY 12189-4000
1	PM TIMS, PROFILER (MMS-P) AN/TMQ-52 ATTN B GRIFFIES BUILDING 563 FT MONMOUTH NJ 07703
1	COMMANDER RADFORD ARMY AMMO PLANT ATTN SMCAR QA HI LIB RADFORD VA 24242-0298
1	US ARMY INFO SYS ENGRG CMND ATTN AMSEL IE TD A RIVERA FT HUACHUCA AZ 85613-5300
1	COMMANDER US ARMY RDECOM ATTN AMSRD AMR W C MCCORKLE 5400 FOWLER RD REDSTONE ARSENAL AL 35898-5000
1	CDR NAVAL RSRCH LAB ATTN TECH LIBRARY WASHINGTON DC 20375-5000
1	US GOVERNMENT PRINT OFF DEPOSITORY RECEIVING SECTION ATTN MAIL STOP IDAD J TATE 732 NORTH CAPITOL ST NW WASHINGTON DC 20402
8	ARMY RSRCH LAB ATTN RDRL WMB D K L MCNESBY ATTN RDRL WMB D A BRANT ATTN RDRL WMB D B HOMAN ATTN RDRL WMB D J COLBURN ATTN RDRL WMB D J RITTER ATTN RDRL WMB D P CONROY ATTN RDRL WMB D R A BEYER ATTN RDRL CIM G T LANDFRIED BLDG 4600 ABERDEEN PROVING GROUND MD 21005-5066
1	US ARMY RSRCH LAB ATTN RDRL ROP TECH LIB PO BOX 12211 RESEARCH TRIANGLE PARK NC 27709-2211
3	US ARMY RSRCH LAB ATTN IMNE ALC HRR MAIL & RECORDS MGMT ATTN RDRL CIM L TECHL LIB ATTN RDRL CIM P TECHL PUB ADELPHI MD 20783-1197

TOTAL: 24 (22 HCS, 1 CD, 1 ELECT)

INTENTIONALLY LEFT BLANK.